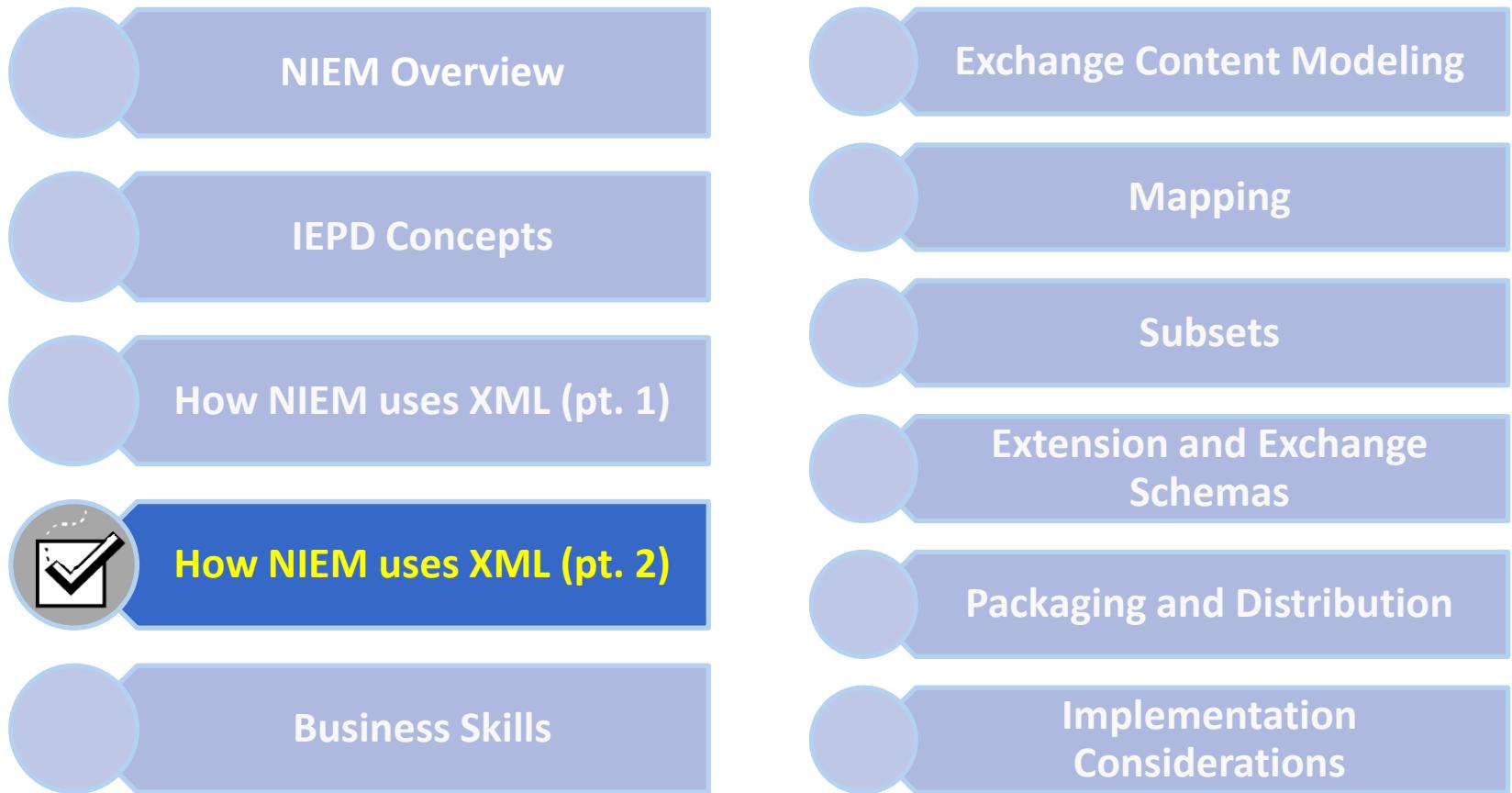


# **How NIEM Uses XML**

## **(part 2)**



# Modules Roadmap: You Are Here



# Objectives Roadmap

This module supports the following course objective:

Describe what NIEM is.

Describe what an IEPD is.

Comprehend artifacts included in an IEPD.

Develop artifacts included in an IEPD.

Package an IEPD.



Understand advanced XML concepts, as required by NIEM.

Recognize business skills required to successfully participate in an IEPD development project.

# Module Objectives

- After completing this module, you should be able to:
  - ◆ Create Substitution Groups.
  - ◆ Create Abstract Elements.
  - ◆ Recognize the concept of Augmentation.
  - ◆ Describe Metadata.
  - ◆ Create Code Tables.

# Substitution Groups

- Some types of information can be represented in multiple ways, for example:
  - ◆ Enumerated values
  - ◆ Date and Time
  - ◆ Many entities can be either a Person or an Organization

# XML Implementation

- NIEM uses XML Schema Substitution Groups to allow a single concept to be represented by multiple elements of different types.
- In the NIEM, Substitution Group heads are usually set as “abstract” and act purely as a placeholders.
- These are “explicit” Groups.

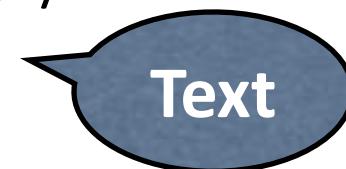
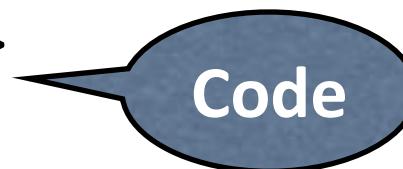
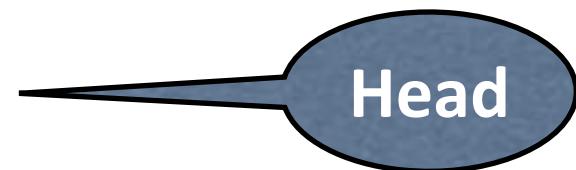
# Implied Substitution Groups

- A few Groups are “implied.”
- For example, **nc:ContactTelephoneNumber**
  - ◆ Is *not* defined as abstract
  - ◆ Can contain a number itself, or
  - ◆ Can be replaced with more specific versions of a Telephone number, such as a **nc:ContactFaxNumber**.
- NIEM tools may not deal with these correctly.



# Schema Example

- <!-- Simplified snippet from the nc: namespace -->
- <xsd:element name="PersonSex" abstract="true"/>
- <xsd:element substitutionGroup="nc:PersonSex" name="PersonSexCode" type="fbi:SEXCodeType"/>
- <xsd:element substitutionGroup="nc:PersonSex" name="PersonSexText" type="nc:TextType"/>



# Instance Example

```
<!-- Code table version -->
<nc:Person>
  <nc:PersonSexCode>
    F
  </nc:PersonSexCode>
</nc:Person>

<!-- Text version -->
<nc:Person>
  <nc:PersonSexText>
    Female
  </nc:PersonSexText>
</nc:Person>
```

# Exercise 11.1: Substitution Groups

- Determine all the ways in which a Date can be represented in NIEM.
  - ◆ Hint: Non-code table Substitution Groups often use the term “Representation” in the Substitution Group head (for example, EntityRepresentation).

# Solution 11.1: Substitution Groups

- nc:DateRepresentation must be substituted by one of these:
  - ◆ nc:Date (A full date)
  - ◆ nc:DateTime (A full date and time)
  - ◆ nc:Year (A year)
  - ◆ nc:YearMonth (A year and month)



# Type Augmentation

- Problem Statement:
  - ◆ Each domain contains objects that other domains might need to use.
  - ◆ Adding these to an object would typically require making a specialization.
  - ◆ But, if objects are needed from multiple domains, then specialization doesn't work.
  - ◆ Cannot extend from multiple base objects.

# One Technique

- Augmentation is meant to solve this problem, allowing objects from multiple domains to be attached to an object.
- Augmentation objects explicitly show that data is just attached to an object without making a special version of that object.
- The end result is more like an object with bags of elements rather than a special type of object.

# Augmentations: *Defined* in NIEM

- Objects from each domain that are ripe for sharing are grouped, for example:
  - ◆ intel:PersonAugmentation
    - (Person-specific from the Intelligence domain)
  - ◆ j:PersonAugmentation
    - (Person-specific from the Justice domain)
  - ◆ em:OrganizationLocationAugmentation
    - (OrganizationLocation-specific from the Emergency Management domain)



# Augmentations: *Used* in an IEPD

- The object being augmented is extended, in an IEPD, to contain the Augmentations holding the needed objects.
- Augmentation groups can be subsetted to contain only those objects desired.
- The difference is more semantic than technical.

# Schema Example

```
<complexType name="IEPDPersonType">
  <complexContent>
    <extension base="nc:PersonType">
      <sequence>
        <element ref="j:PersonAugmentation"/>
        <element ref="intel:PersonAugmentation"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="IEPDPerson" type="local-ns:IEPDPersonType"
  substitutionGroup="nc:Person"/>
```

Defined in  
our IEPD

Added via  
Substitution  
Groups

# Instance Example

```
<local-ns:IEPDPerson>
  <nc:PersonBuildText>Husky</nc:PersonBuildText>
  <!-- More Common Person-related elements -->

  <j:PersonAugmentation>
    <j:PersonDrivingInsuranceStatusText>
      None at all
    </j:PersonDrivingInsuranceStatusText>
    <!-- More Justice Person-related elements -->
  </j:PersonAugmentation>

  <intel:PersonAugmentation>
    <intel:OccupationName>
      NIEM Trainer
    </intel:OccupationName>
    <!-- More Intel Person-related elements -->
  </intel:PersonAugmentation>
</local-ns:IEPDPerson>
```



# Exercise 11.2: Augmentation

- Suppose your exchange requires a Person object with the following properties:
  - ◆ Person Name
  - ◆ Hair Color
  - ◆ FBI ID Number
  - ◆ Person Function
- Which augmentation element(s) would be used to extend the base Person object?

# Solution 11.2: Augmentation

- j:PersonFBIIdentification is contained in j:PersonAugmentation
- it:PersonFunctionText is contained in it:PersonAugmentation
- Person Name and Hair Color are already present via the nc: namespace



# Metadata

- Metadata is data about other data.
- For example, a Protection Order contains information about the people involved, timeframes, jurisdiction, and more.
- There is also information about the Protection Order, such as the language in which the Order itself is written.
- Uses include searching, categorizing, and classifying.

# Fuzzy Lines

- The dividing line between metadata and data is sometimes unclear.
- Whether some piece of data is metadata often relies on context.
- If it feels like metadata, then treat it as metadata.
- If it feels like data, then treat it as data.

# XML Implementation

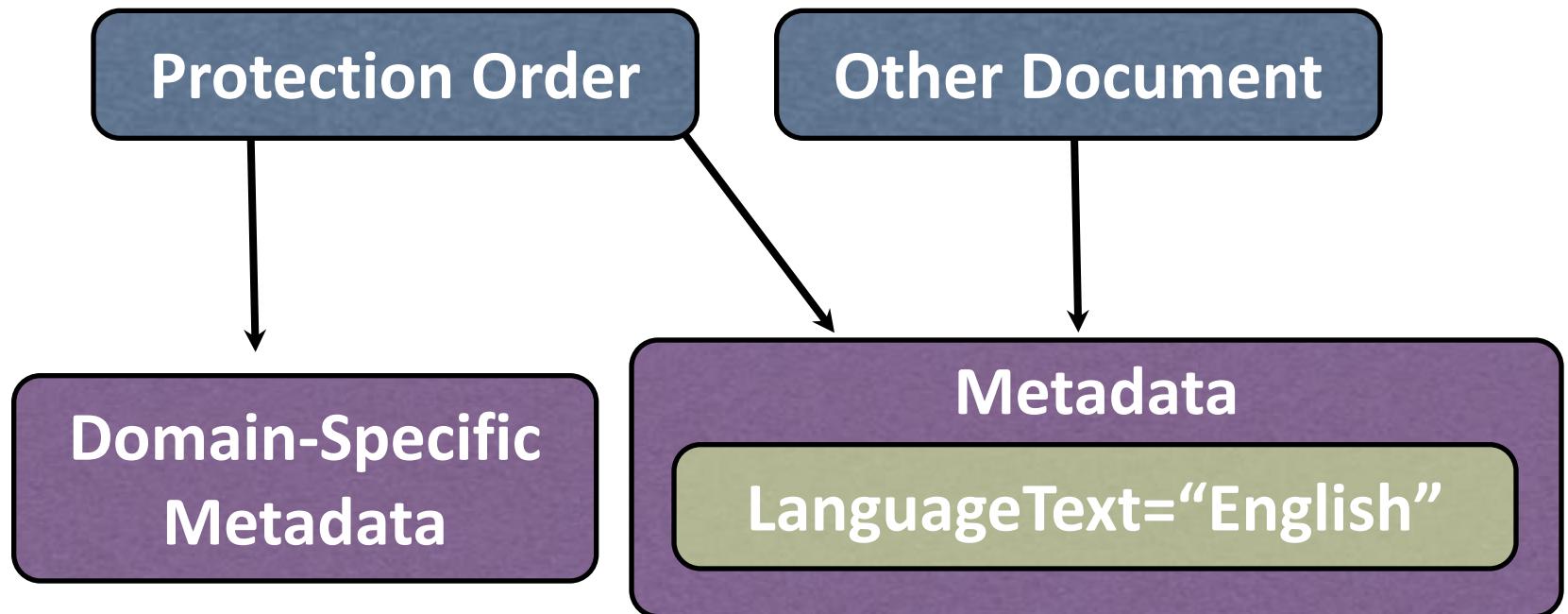
- NIEM uses dedicated Metadata objects to hold structured metadata information.
- **nc:Metadata** holds general purpose Metadata. There are also specific types of Metadata for some domains and some types of objects.
- Metadata can be extended like other NIEM objects.



# Flexibility

- NIEM objects point to the applicable Metadata.
  - ◆ Uses a space-delimited series of IDs.
- NIEM objects can point to more than one Metadata object.
- Multiple NIEM objects can point to the same Metadata object.
- NIEM objects at any level can have Metadata.

# Metadata Combinations



Not all combinations will make sense.

# Schema Example

```
<xsd:complexType name="MetadataType">
  <xsd:complexContent>
    <xsd:extension base="s:MetadataType">
      <xsd:sequence>
        <!-- Additional metadata properties -->
        <xsd:element ref="nc:LanguageText"/>
        <!-- Additional metadata properties -->
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Metadata" type="nc:MetadataType"/>

<!-- j:ProtectionOrder defined elsewhere -->
```



# Instance Example

```
<nc:Metadata s:id="ID0002">
  <nc:LanguageText>
    English
  </nc:LanguageText>
</nc:Metadata>

<j:ProtectionOrder s:metadata="ID0002">
  <j:ProtectionOrderRestrictedPerson>
    <nc:PersonName>
      <nc:PersonFullName>
        I. M. Protected
      </nc:PersonFullName>
    </nc:PersonName>
  </j:ProtectionOrderRestrictedPerson>
</j:ProtectionOrder>
```



# Code Tables

- Code tables are common throughout government.
- Code tables improve information sharing by constraining possible values.
- Code tables can be very large and / or continually updated.
- NIEM is not an authoritative source for any code tables.

# XML Implementation

- NIEM encapsulates code tables into separate namespaces, based on authoritative sources.
  - ◆ For example, NCIC, ISO, FIPS, FBI, ANSI-NIST, and many more.
- This allows these authoritative bodies to eventually take over maintenance of the code tables.

# Restricted Enumerations

- Code tables are represented by starting with the token data type and restricting it to a number of enumerations.
- Code tables cannot be expanded. To add additional codes, either:
  - ◆ Make a copy of the code table, adding the additional codes, or
  - ◆ Create a second code table with the additional codes and make it substitutable for the existing table via Substitution Groups.

# Schema Example

```
<xsd:simpleType name="USStateCodeSimpleType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="MN">
      <xsd:annotation>
        <xsd:documentation>MINNESOTA</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="USStateCodeType">
  <xsd:simpleContent>
    <xsd:extension base="usps:USStateCodeSimpleType">
      <xsd:attributeGroup ref="s:SimpleObjectAttributeGroup"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!-- Simplified snippet from the nc: namespace -->
<xsd:element name="LocationStateUSPostalServiceCode"
  type="usps:USStateCodeType"/>
```

# Instance Example

```
<!-- Simplified snippet from the usps: namespace -->
```

```
<nc:LocationStateUSPostalServiceCode>
```

MN —————

```
</nc:LocationStateUSPostalServiceCode>
```

Code  
values are  
validated

# Exercise 11.3: Code Tables

- How many namespaces define a code for the state of Minnesota?
- What are the types in each of those namespaces?
  - ◆ Hint: Search for Facets in the NIEM SSGT, or use Wayfarer.

# Solution 11.3: Code Tables

9 code tables defined in 5 namespaces

Namespaces	Code Tables
ansi_d20	JurisdictionAuthorityCodeSimpleType
fips_10-4	InternationalStateCodeSimpleType
fips_5.2	USStateCodeSimpleType USStateNumericCodeSimpleType
ncic	LISCodeSimpleType LSTACodeSimpleType POBCodeSimpleType RESCodeSimpleType
usps_states	USStateCodeSimpleType



# Module Summary

- After completing this module, you should be able to:
  - ◆ Create Substitution Groups.
  - ◆ Create Abstract Elements.
  - ◆ Recognize the concept of Augmentation.
  - ◆ Describe Metadata.
  - ◆ Create Code Tables.

## Creative Commons



### **Attribution-ShareAlike 2.0**

#### **You are free to**

- Copy, distribute, display, and perform the work
- Make derivative works
- Make commercial use of the work



**Attribution**—You must give the original author credit



**ShareAlike**—If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one

#### **Under the following conditions**

- For any reuse or distribution, you must make clear to others the license terms of this work
- Any of these conditions can be waived, if you get permission from the copyright holder

#### **Your fair use and other rights are in no way affected by the above**

This is a human-readable summary of the [Legal Code \(the full license\)](#) and [Disclaimer](#)

This page is available in the following languages

[Català](#), [Deutsch](#), [English](#), [Castellano](#), [Suomeksi](#), [français](#), [hrvatski](#), [Italiano](#), [日本語](#), [Nederlands](#), [Português](#), and [中文\(繁\)](#)

[Learn how to distribute your work using this license](#)